利用 Logisim 开发 MIPS 单周期处理器

一、整体结构：

控制器（Controller）、IFU（取指令单元）、GRF（通用寄存器组，也称为寄存器文件、寄存器堆）、ALU（算术逻辑单元）、DM（数据存储器）、EXT（位扩展器）。

处理器为 32 位处理器

处理器应支持的指令集为：{addu, subu, ori, lw, sw, beq, lui, nop}。

二、模块规格：

（一）模块规格撰写

1.IFU（取指令单元）：内部包括 PC、IM、nPC

1）PC（程序计数器）

器件：32bit 寄存器

2）IM（指令存储器）

器件：32*32bitROM

因为 ROM 中储存了 32 个地址，且 IM 实际地址宽度仅为 5 位，所以在 PC 正在读取的地址端口采用两个对接的 Splitter 器件，从而将地址的低 5 位连接到 ROM 选择地址端口。

3）nPC（计算下一条指令）

器件：加法器、1 位多路选择器

在处理器支持的指令集中，下一条指令的选择分为两种情况：

1）addu, subu, ori, lw, sw, lui, nop：PC=PC+4

2）beq：PC=(PC+4）+sign_extend(imm16)<<2

模块接口

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：有效

		0：无效
PCSrc	I	PC 选择信号 1：当前指令为 beq 0：当前指令不为 beq
Zero	I	ALU 计算结果为 0 标志 1：计算结果为 0 0：计算结果非 0
Instr[31:0]	O	32 位 MIPS 指令

功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时，PC 被设置为起始地址 0x00000000
2	取指令	根据 PC 从 IM 中取出指令
3	计算下一条指令地止	PCSrc=0 时：PC=PC+4 PCSrc=1 时：PC=(PC+4）+sign_extend(imm16)<<2

2.GRF（通用寄存器组）：内部包括 32 个寄存器

具有写使能的寄存器实现，寄存器总数为 32 个

0 号寄存器的值始终保持为 0。其他寄存器初始值均为 0

模块接口

信号名	方向	描述
RegWrite	I	读写控制信号 1：写操作 0：读操作
clk	I	时钟信号
reset	I	复位信号 1：有效 0：无效

Read_reg1	I	读寄存器 1 的地址
Read_reg2	I	读寄存器 2 的地址
Write_reg	I	写寄存器的地址
Read_data1	O	32 位输出 1
Read_data2	O	32 位输出 2

功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时，所有寄存器的值被设置为 0x00000000
2	写寄存器	根据输入的写寄存器地址，把输入的数据写入写寄存器中
3	读寄存器	根据输入的读寄存器地址，将数据读出

3.ALU（**算术逻辑单元**）

提供 32 位加、减、或运算

可以不支持溢出

模块接口

信号名	方向	描述
A[31:0]	I	ALU32 位输入数据 A
B[31:0]	I	ALU32 位输入数据 B
ALUop[1:0]	I	ALU 功能选择信号 00:加法 01:减法 10:或运算
Result	O	32 位数据输出

功能定义

序号	功能名称	功能描述
1	或	A|B

2	减	A-B
3	加	A+B

4.DM（数据存储器）

使用 RAM 实现，容量为 32bit*32, 采用双端口模式

起始地址：0x00000000

RAM 应使用双端口模式，即设置 RAM 的 Data Interface 属性为 Separate load and store ports

模块接口

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：有效 0：无效
Address[4:0]	I	操作寄存器地址
WData[31:0]	I	写入数据的输入
RData	O	读取数据的输出
MemRead	I	读写控制信号 1：读操作
MemWrite	I	读写控制信号 1：写操作

功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时,所有数据被设置为 0x00000000
2	写操作	根据输入的寄存器地址，把输入的数据写入
3	读操作	根据输入的寄存器地址，将其中的数据读出

5. EXT（位扩展器）：

使用 logisim 内置的 Bit Extender

模块接口

信号名	方向	描述
Imm[15:0]	I	16 位 imm 数据输入
Out[31:0]	O	位扩展后的 32 位输出
Extop[1:0]	I	位扩展选择信号 00:高位补 0 01：高位符号扩展 10：低位补 0
Result	O	32 位数据输出

功能定义

序号	功能名称	功能描述
1	高位补 0	高 16 位补 0
2	低位补 0	低 16 位补 0
3	符号扩展	若符号位为 0，则高位补 0 若符号位为 1，则高位补 1

6.控制器（Controller）

模块接口

信号名	方向	描述
op[5:0]	I	6 位 op
func[5:0]	I	6 位 function
RegDst	O	写寄存器地址控制
ALUsrc	O	ALU 的 B 操作数的选择控制
Memtoreg	O	写寄存器的数据来源选择控制
RegWrite	O	GRF 读写控制信号

MemWrite	O	DM 写控制信号，写入 GRF 的数据选择
PCsrc	O	PC 选择信号
Extop	O	控制扩展方式
ALUSrc[1:0]	O	控制 ALU 进行相应运算

（二）思考题

1. 若 PC（程序计数器）位数为 30 位，试分析其与 32 位 PC 的优劣。

答：当使用 32 位 PC 时，在 MIPS 编写指令时，认为指令在内存中以字节为单位进行存储，取下一条指令时，应+4；对于 beq 指令，在 PC+4 之后，需要对 offet 左移两位。因此 32 位 PC 每次+4 后，取其 2 至 6 位作为地址输入 ROM 中。这种方式比较易于理解，符合 MIPS 的指令表达式

当使用 30 位 PC，在 ROM 中存储指令时，以字为单位进行存储，取下一条指令时，应+1。直接取 0 至 4 位作为地址输入 ROM 中。这种方式不需要将 offset 左移两位，比较简便。

2. 现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 使用寄存器，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出

答：合理，无改进意见；

IM 为指令寄存器，以取指令为主，指令一旦写入便不再改变；

DM 为数据寄存器，既存又取，需要能够随时写入与读出；

GRF 为通用寄存器组，应该由 32 个寄存器组成，可存可取。

（三）控制器设计

（一）设计方式

1. 数据通路设计

指令	Adder		PC	IM	Registers				ALU		DM		Ext	Nadd	
	A	B		Address	Reg1	Reg2	Wreg	Wdata	A	B	Address	Wdata			
addu/subu	PC	4	Adder	PC	Rs	Rt	Rd	ALU	Rdata1	Rdata2					
Ori	PC	4	Adder	PC	Rs		Rt	ALU	Rdata1	Ext			imm16		
Lw	PC	4	Adder	PC	Rs		Rt	DM	Rdata1	Ext	ALU		imm16		
Sw	PC	4	Adder	PC	Rs	Rt			Rdata1	Ext	ALU	Rdata2	imm16		
Beq	PC	4	Adder|Nadd	PC	Rs	Rt			Rdata1	Rdata2			imm16	Adder	Shift
Lui	PC	4	Adder	PC	$0		Rt	ALU	Rdata1	Ext			imm16		
Nop															
合并	PC	4	Adder|Nadd	PC	Rs	Rt	Rt|Rd	ALU|DM	Rdata1	Rdata2|Ext	AlU	Rdata2	imm16	Adder	Shift

2. 主控单元真值表

I or O	Signal name	Addu /subu	ori	lw	sw	beq	lui
Input	Op0	0	1	1	1	0	1
	Op1	0	0	1	1	0	1
	Op2	0	1	0	0	1	1
	Op3	0	1	0	1	0	1
	Op4	0	0	0	0	0	0
	Op5	0	0	1	1	0	0
Output	RegDst	1	0	0	X	X	0
	ALUSrc	0	1	1	1	0	1
	MemtoReg	0	0	1	X	X	0
	RegWrite	1	1	1	0	0	1
	MemRead	0	0	1	0	0	0
	MemWrite	0	0	0	1	0	0
	PCSrc	0	0	0	0	1	0
	ExtOp[1:0]	xx	00	01	01	01	10
	ALUOp[1:0]	00/01	10	00	01	00	10

（二）思考题

1.结合上文给出的样例真值表，给出 RegDst， ALUSrc， MemtoReg，RegWrite，nPC_Sel，ExtOp 与 op 和 func 有关的布尔表达式（表达式中只能使用“与、或、非”3 种基本逻辑运算。）

$$\text{RegDst} = (\overline{Op}0\&\overline{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{ALUSrc} = (\text{Op}0\&\overline{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\text{Op}3\&\overline{Op}4\&Op5)|(\text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{MemtoReg} = \text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5$$

$$\text{RegWrite} = (\overline{Op}0\&\overline{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5)|(\text{Op}0\&\overline{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)|(\text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{PCSrc} = \overline{Op}0\&\overline{Op}1\&\text{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5$$

$$\text{ExtOp}[1] = \text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5$$

$$\text{ExtOp}[0] = (\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\text{Op}3\&\overline{Op}4\&Op5)|\overline{Op}0\&\overline{Op}1\&\text{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5$$

2.充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式，请给出化简后的形式。

$$\text{RegDst} = (\overline{Op}0\&\overline{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{ALUSrc} = (\text{Op}0\&\overline{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\text{Op}3\&\overline{Op}4\&Op5)|(\text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{MemtoReg} = \text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5$$

$$\text{RegWrite} = (\overline{Op}0\&\overline{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5)|(\text{Op}0\&\overline{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)|(\text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5)$$

$$\text{PCSrc} = \overline{Op}0\&\overline{Op}1\&\text{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5$$

$$\text{ExtOp}[1] = \text{Op}0\&\text{Op}1\&\text{Op}2\&\text{Op}3\&\overline{Op}4\&\overline{Op}5$$

$$\text{ExtOp}[0] = (\text{Op}0\&\text{Op}1\&\overline{Op}2\&\overline{Op}3\&\overline{Op}4\&\text{Op}5)$$
$$|(\text{Op}0\&\text{Op}1\&\overline{Op}2\&\text{Op}3\&\overline{Op}4\&Op5)|\overline{Op}0\&\overline{Op}1\&\text{Op}2\&\overline{Op}3\&\overline{Op}4\&\overline{Op}5$$

（四）测试 CPU

（一）测试代码

```
2   .text
3     ori $t0, $0, 3
4     ori $t1, $0, 2
5     addu $t2, $t0, $t1
6     subu $t3, $t0, $t1
7     sw $t2, 0($sp)
8     lw $t4, 0($sp)
9     lui $t5, 1
10    beq $t2, $t4, jump
11    addu $t6, $0, $0
12    jump:
13    addu $t6, $0, $t0
```

Address	Code	Basic		
0x00003000	0x34080003	ori $8, $0, 0x00000003	3:	ori $t0, $0, 3
0x00003004	0x34090002	ori $9, $0, 0x00000002	4:	ori $t1, $0, 2
0x00003008	0x01095021	addu $10, $8, $9	5:	addu $t2, $t0, $t1
0x0000300c	0x01095823	subu $11, $8, $9	6:	subu $t3, $t0, $t1
0x00003010	0xafaa0000	sw $10, 0x00000000($29)	7:	sw $t2, 0($sp)
0x00003014	0x8fac0000	lw $12, 0x00000000($29)	8:	lw $t4, 0($sp)
0x00003018	0x3c0d0001	lui $13, 0x00000001	9:	lui $t5, 1
0x0000301c	0x114c0001	beq $10, $12, 0x00000001	10:	beq $t2, $t4, jump
0x00003020	0x00007021	addu $14, $0, $0	11:	addu $t6, $0, $0
0x00003024	0x00087021	addu $14, $0, $8	13:	addu $t6, $0, $t0

2. 预期结果

序号	Instr	RegWrite	RegAddr	RegData	MemWrite	MemAddr	MemData
3	0x34080003	1	5'b01000	3	0	5'b00000	0x00000000
4	0x34090002	1	5'b01001	2	0	5'b00000	0x00000000
5	0x01095021	1	5'b01010	5	0	5'b00001	0x00000002
6	0x01095823	1	5'b01011	1	0	5'b00000	0x00000002
7	0xafaa0000	0	5'b01010	0	1	5'b00000	0x00000005
8	0x8fac0000	1	5'b01100	5	0	5'b00000	0x00000000
9	0x3c0d0001	1	5'b01101	65536	0	5'b00000	0x00000000
10	0x114c0001	0	5'b01100	10	0	5'b00010	0x00000005
11	0x00007021	不执行					
13	0x00087021	1	5'b01110	3	0	5'b00000	0x00000003

（三）思考题

1. 前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM 片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移。

答：即选择 ALU 32 位输出中的的低 2 至 6 位作为 DM 的 5 位地址信号。

2.除了编写程序进行测试外，还有一种验证 CPU 设计正确性的办法——形式验证。 形式验证的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证（Formal Verification)”了解相关内容后，简要阐述相比与测试，形式验证的优劣。

答：所谓形式验证，是指从数学上完备地证明或验证电路的实现方案是否确实实现了电路设计所描述的功能。形式验证方法分为等价性验证、模型检验和定理证明等。形式验证时要确定电路在哪一级电路上的测试是正确的，使用模型检验的方法看两个电路在描述上是否一致。

	优点	缺点
仿真测试	操作速度快	初始设置时间长，debug 较为困难，验证周期长
形式验证	易于发现细节处的 bug	所需数学知识多，分布小规模，仅在验证小模块时较为实用

附表：

	addu	subu	ori	lw	sw	beq	lui
op	000000	000000	001101	100011	101011	000100	001111
func	100001	100011					
RegDst	01	01	00	00	00	00	00
ALUSrc	0	0	1	1	1	0	1
MemtoReg	00	00	00	01	00	00	00
RegWrite	1	1	1	1	0	0	1
MemWrite	0	0	0	0	1	0	0
PCSrc	00	00	00	00	00	01	00
Extop	xx	xx	00	01	01	01	10
ALUop	00	01	10	1	00	01	10
	jal	j	sao	bgez			
op	000011	000010	000000	100001			
func			011111				
RegDst	10	x	1	x			
ALUSrc	x	x	0	0			
MemtoReg	10	x	0	0			
RegWrite	1	0	1	0			
MemWrite	0	0	0	0			
PCSrc	10	10	0	1			

Extop	xx	xx	xx	xx			
ALUop	xx	xx	11	xx			